

جامعة القدس المفتوحة - غزة
برنامج التكنولوجيا والعلوم التطبيقية
تخصص أنظمة المعلومات الحاسوبية

برنامج يقوم برسم بعض الأشكال الهندسية
بأسلوب البرمجة الكينونية

إعـداد الطـالــب
محمد علي أبوحجر

2005م – 1426هـ

بسم الله الرحمن الرحيم

الاسم : محمد علي أبو حجر
رقم جامعي : 2009005312

- **اسم البرنامج :-**

رسم أشكال هندسية .

- **الأهداف المطلوبة :-**

إعداد برمجية تعليمية محوسبة (كتطبيق عملي لمادة هندسة البرمجيات وورشها) ، بحيث تخضع هذه البرمجية للمعايير والمقاييس المتبعة في بناء برمجيات تؤدي المهام المطلوبة منها بكفاءة وفاعلية وتحقق خصائص البرمجيات المتعارف عليها بين مراكز صناعة البرمجيات ما أَمكن .

- **مفهوم هندسة البرمجيات :-**

هي بناء و إنتاج أنظمة برمجية حاسوبية وفق الأسس العلمية الحديثة ، تتميز بالجودة العالية وبتكلفة منخفضة وتنجز في وقت قصير ، وتشتمل على مجموعة من الأساليب والإجراءات التي تضمن إنجاز عمليات التحليل والتصميم والبرمجة والاختبار بكفاءة وفاعلية

- **صفات البرمجية الجيدة :-**

1. البرمجيات الاقصر في الطول هي الافضل .
2. القرارات الاقل هي الافضل

من الناحية المنطقية فان تصميم البرمجيات الذي يحوي اقل عدد من نقاط القرارات يكون افضل من الذي يحوي عددا اكبر ، وان كفاءة اي برنامج تتناسب عكسيا مع عدد نقاط القرارات والتشعبات .

3. تجنب تداخل القرارات .

التداخل في القرارات من علامات ضعف التصميم ، لا تحاول لن تجعل التصميم يزيد عن ثلاث مستويات .

4. تركيب البيانات المستخدم يجب ان يحدد بصورة جيدة .

اختيار تركيب البيانات المناسب .

5. إضافة اكبر قدر من الشرح والتوضيح .
6. الانسجام والتكامل .

يقصد به ان يتم تصميم فقرات البرمجيات باسلوب موحد ونمطي وان يتم ربطها من خلال روابط البيانات التي تضمن استقلالية ثلاث الوحدات الوظيفية .

7. الكفاءة والفاعلية .

يقصد بالكفاءة استخدام اقل حجم ممكن من الموارد (الذاكرة - وحدات التخـزين) للحصـول علي سرعة في التنفيذ ، أما الفاعلية فهي قيام البرمجية بالوظائف المطلوبة منها وفقا لرغبـة واحتياجات المستخدم .

خطة العمل :-

لقد وقع اختياري على كتابة برنامج بلغة ++c تقوم برسم مجموعة من الأشكال الهندسية الأساسـية مثل الخط والمثلث والاسطوانة .. الخ ، مستخدما مفاهيم البرمجة الكينونيـة كأحـد أسـاليب هندسـة البرمجيات في بناء النظم البرمجية وتطويرها التي تتمثل في الوراثة والأصناف المشتقة وخاصية تعدد الأوجه والميزات الأخرى 1التي توفرها لغة ++c للوصول إلى برمجية تتمتع بالكفـاءة والعموميـة ودرجة عالية من البساطة والوضوح والتسلسل المنطقي .

ولكي اصل إلى ذلك الشيء قمت باتباع مجموعة الخطوات اللازمة لبناء أنظمة كيانات وهي :

- تحديد تركيب الكيان المطلوب .
- تحديد المتغيرات التي يتضمنها الكيان .
- تحديد البرامج الفرعية المطلوبة لتنفيذ عمليات الكيان .
- تحديد أهداف كل برنامج فرعي ، وأهمية المتغيرات الأساسية ، وتحديد الشروط الأولية والنتائج المتوقعة .

لماذا تم اختيار الكيانات

- تعد برمجة الكيانات Object Oriented Programming اتجاها جديدا في البرمجـة ، فهـو وسيلة تساعد المبرمجين في معالجة البرامج الكبيرة وتسهيل تتبع البرامج المعقـدة ، ويعكـس مفهـوم الاشياء الموجهة إلى إمكانية تنظيم البرمجيات لمجموعة من الاشياء المتقطعـة Discrete Objects المتكونة من توليفة من هياكل البيانات Data Structure والسلوك Behavior حيث ان هذا الاتجاه الجديد في البرمجة يحاول تقسيم الاشياء إلى اصناف وتشترك اعضاء كل صنف في الصفات والسلوك ويمكن للمبرمج ان يبني مكتبة من الاصناف او يستعين باخرى جاهزة ، ويحتوي كـل صـنف علـي البيانات فضلا عن سلوكها او العمليات المنتظر إجراؤها عليها بحيث تشكل معا نمطا ونوعا حيـا مـن سلوكها .

- تعد برمجة الكيانات اسلوبا من اساليب هندسة البرمجيات التي توفر إمكانيات متعددة منها :

- التنظيم الجيد والمقاطع البرمجية المتكاملة .
- الوضوح والانسجام بين مقاطع المنتج .
- الكفاءة والاستخدام المتعدد .

- والكيان اسلوب جديد في التفكير في بناء البرامج حيث يجمع حقول البيانات مع البرامج الفرعية ، التي تسند في عملها على تلك الحقول في بنية واحدة .

- يتم التعامل مع الكيان وكأنه وحدة واحدة ، وهذه الوحدة مستقلة بذاتها من حيت تركيب البيانات والعمليات المطلوبة .

- والكيان عبارة عن وسيلة أو كينونة لها استقلاليتها تقوم بأداء مجموعة مهام خاصة بها .

- وتتصف الكيانات بثلاث صفات هي : البنية الواحدة والتوارث والاستعمال المتعدد .

مميزات برمجة الكيانات

1. تصميم برمجيات ذات متانة عالية ، وقابلة للاستخدام المتعدد ، وتتميز بخاصية التوارث .
2. إمكانية إضافة مقاطع برمجية جديدة لن يكون لها تأثير مباشر على بقية الوحدات ولن تؤدي إلى حدوث أخطاء .
3. إمكانية حذف وحدات قديمة دون الحاجة إلى تعديل متعلقاتها .
4. إمكانية التعديل فقط على مقطع معين وسينتقل التأثير تلقائيا إلى بقية أجزاء البرنامج .
5. تعتبر وسيلة عرض للتصميم ، تجعل الاستفادة من الفعاليات والأنشطة أمرا مرغوبا ومستحسنا.

المفاهيم البرمجية التي تم تطبيقها :-

1. الوراثة . Inheritance
2. الأصناف المشتقة . Derived Classes
3. تعدد الأوجه . Polymorphism
4. جمل الدوران والانتقاء . Looping & Selection Statements
5. جمل الإدخال والإخراج وبعض الو سوم Tags .
6. دوال لرسم الأشكال . Graphic Functions

القدرات التي تمتع بها البرمجية المنفذة :-

1. جمل تعليمات ومساعدة .
2. وجود عنوان يتصدر البرمجة .
3. البرمجية تتقبل الاسم الشخصي للمستخدم لتكون المخاطبة فيما **بعد شخصية .**
4. المستخدم غير مقيد في عرض البرمجية ويمكنه الانتهاء والخروج من البرنامج متي شاء .
5. تم تطبيق مفهوم الوراثة والاشتقاق وخاصية تعدد الأوجه .

المعلومات الأساسية :-

1. الأصناف المشتقة والوراثة . Derived Classes & Inheritance

a. تمكننا لغة ++c من تعريف صنف جديد مشتق من صنف اخر والفائدة من عمل ذلك الصنف هو تقليل الحاجة إلي كتابة دوال جديدة للصنف الجديد ، وذلك لاننا نستطيع اســتخدام الــدوال

الخاصة للصنف الأساس للصنف المشتق مما يقلل في الوقت اللازم لصيانة هذه الدوال ، والمساعدة في إنتاج دوال أكثر كفاءة و عمومية More General & Accurate .

b. تنظم الأصناف المختلفة بطريقة هرمية تضمن أن تعرف المتغيرات والدوال التي تستخدم للعديد من الأصناف المشتقة في اعلي الشجرة ، بينما تقع الأصناف المشتقة والتي لا تستخدم في تعريف أصناف جديدة في أسفل الشجرة الهرمية.

ولعمل ذلك لا بد من دراسة وفهم النظام المطلوب وتطويره وإعداده .

في حالتنا :

التنظيم الهرمي الأتي يوضح الأصناف المشتقة والصنف الأساس لتمرين رسم أشكال .

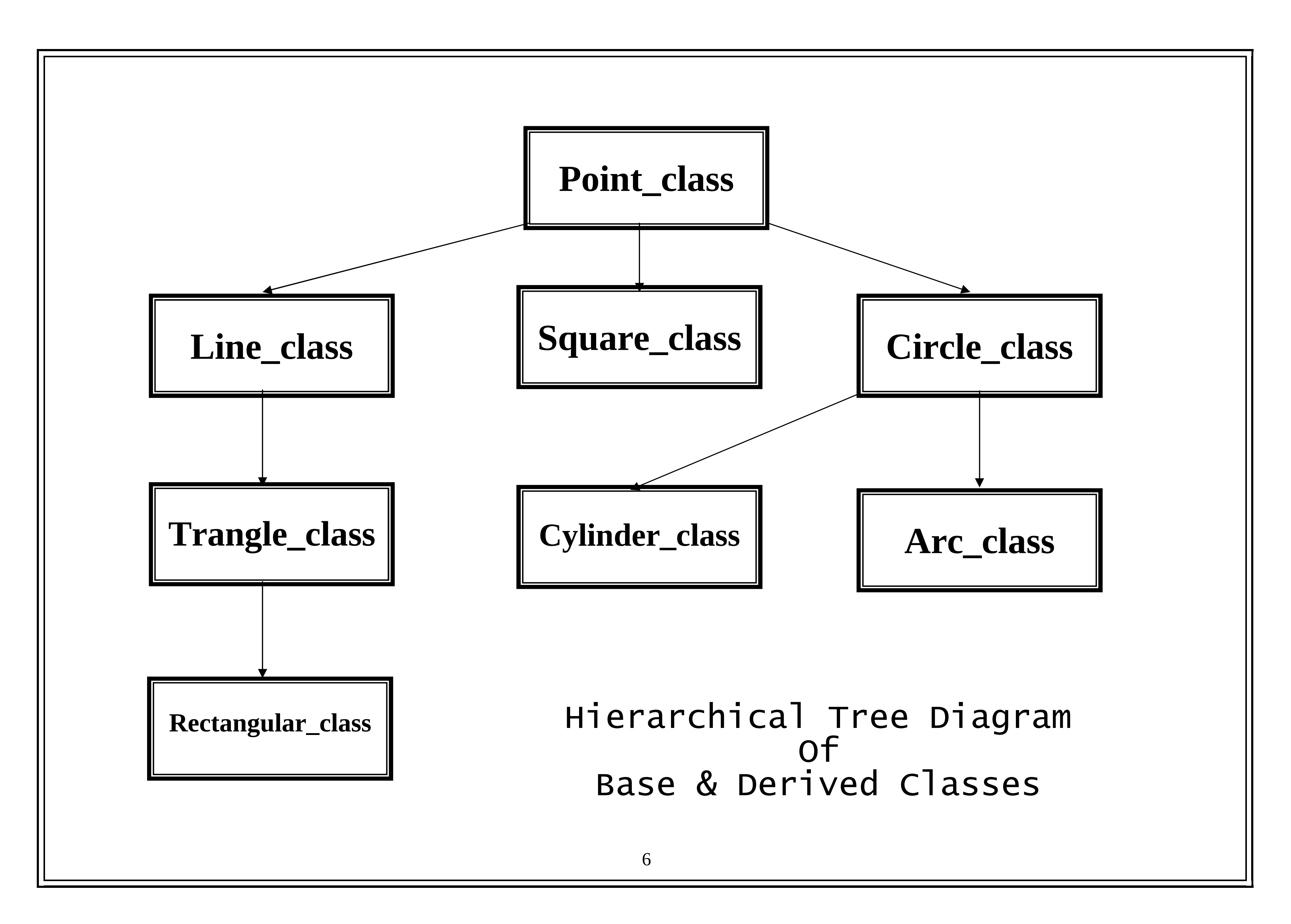
هذه الشجرة تخبرنا بان هناك صنف أساسيا هو point_class وان هناك ثلاثة أصناف مشتقة منه هي line_class و square_class و circle_class ، والصنف المشتق line_class مشتق منه أيضا صنف جديد يسمى trangle_class وان هذا الصنف يشتق منه صنف أخر يسمى rectangular_class ، وان الصنف circle_class يشتق منه صنفين هما cylinder_class وكذلك الصنف arc_class .

c. نوع الوراثة المستخدمة (وراثة عامة) Public inheritance وهي تعني:

a. المتغيرات والدوال المعرفة في القسم الخاص Private من الصنف الأساس تصبح غير قابلة للمعالجة من قبل الدوال المنتمية للصنف المشتق .

b. المتغيرات المعرفة في القسم المحمي Protected من الصنف الاساس تبقي محمية بالنسبة للصنف المشتق .

c. المتغيرات والدوال المعرفة في القسم العام Public من الصنف الاساس تبقى عامة بالنسبة للصنف المشتق .


Point_class
Line_class
Square_class
Circle_class
Trangle_class
Cylinder_class
Arc_class
Rectangular_class
Hierarchical Tree Diagram
Of
Base & Derived Classes

2. تعدد الأوجه Polymorphism

1. تعدد الأوجه من الميزات التي تقمها لغة C++ فهي تمنحنا القدرة علي تأجيل تحديد هوية الدالة المراد استدعاها من وقت الترجمة Compiler Time التي وقت التنفيذ Run Time (عند التنفيذ فقط سنعرف طبيعة القيم المخزنة في المصفوفة .
2. كما تمكننا لغة C++ من وضع عنوان لكائن من صنف مشتق في متغير مؤشر معرف علـــي انه مؤشر للصنف الأساس .

في حالتنا :

```
Point_class *A[8]
.....
.....
A[i]=new circle_class
```

ولعرض البيانات الخاصة بالكائنات نستدعي الدالة draw()

```
For(int j=0;j<=7;j++)
        A[j]->draw();
```

3.لإخبار مترجم اللغة C++ بان يؤجل تحديد هوية الدالة draw() الـــ وقت تنفيذ البرنامج لا بد من تعريف الدوال في جميع الاصناف على انها دوال وهمية Virtual Functions

3. التوارث Inheritance

تعني إمكانية الاستفادة من التعاريف والبرامج الموجودة في كيانات سابقة عند تصميم بنية جديدة ، حيث يمكن للبنية الجديدة ان ترث بعض او كل الخصائص في البرامج السابقة ، وتوفر خاصية التوارث الوقت والجهد .

4. الكيانات الديناميكية

بالنسبة للكيانات الديناميكية فهي تؤدي دورا بارزا في بناء هندسة البرمجيات ، حيث يتم تخصيص جزء من الذاكرة للكيانات خلال مرحلة تنفيذ البرامج ويلغي التخصيص ويعاد حجم الذاكرة المقتطع إلى وضعه الاصلي حال انتهاء دور الكيان ويوفر هذا الاسلوب إمكانيات كبيرة لمهندس البرمجيات .

New لحجز موقع في الذاكرة

Delete لإلغاء عملية الحجز ، ويحرر الذاكرة من البرنامج

جمل البرمجية :- (شرح المقاطع المستخدمة)

الجزء التالي يقدم معلومات عن كاتب البرمجية والمؤسسة التابع لها ولغة البرمجة المستخدمة

```
/* NAME OF PROGRAM : SHAPES - LANGUAGE USE : C++  */
/* AUTHOR : MOHAMMED ABU HAJAR - ID :2009005312 */
/* ACADEMY SUPERVISOR : DR. SANA WAFAA SAIGH */
/* UNIVERSITY : OPEN QODS UNIVERSITY , GAZA REGION EDUCATION */
```

الجزء التالي يبين المكتبات التي تم استدعاؤها

```
#include <iostream.h>
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
```

الجزء التالي يبين تعريف بعض الدوال الفرعية والمتغير name

```
void initialize();
void main_menu(); // menu which we select the choice
char name[30];    // declare for name as array of char
```

الجزء التالي يعرف الكائن point – حيث يعتبر الصنف الرئيسي - الذي يتكون من المتغيرين x1 وy1 ودالة البناء point_class ودالة الرسم draw

```
class point_class //  Base Class
{
 protected:
   int x1; // the first point
   int y1; // the second point
 public:
   point_class()  // Constructor
    { /* inline function */
      cout<<"\n\t"<<name<<" Enter X Coordinates (X1) = "; cin>>x1; // get first point
      cout<<"\n\t"<<name<<" Enter Y Coordinates (Y1) = "; cin>>y1; // get second point
    }
   virtual void draw();   // Print Function
};
void point_class :: draw()// details of point class
 {
  clearviewport();        // Clear View Point of drawing area
  for(int i=1;i<=3;i++)   // to draw in three line
   {
    for(int j=0;j<=350;)  // Delay for 500ms
      {
        putpixel(x1+j,y1,i+12);   // drawing the points
        for(long c=1;c<=8000000;c++)  { }
        j=j+50;  // next step
      }
        y1=y1+50; // Next
   }
 }
```

الجــزء الـتــالي يـعــرف الصــنف circle_class حيــث يــرث مــن الصــنف الـرئـيســي point_class ونـوع الـوراثــة عامــة ، ويـتكــون مــن المــتغير radius والدالــة المنتمية draw

```
class circle_class : public point_class  // Circle Class
{                    // Drived From point_class
 protected:
   int radius;  // define the radius as integer number
 public:
   circle_class()
    {
     // point_class :: point_class();
     cout<<"\n\t"<<name<<" Enter Radius = ";
     cin>>radius;  /* get the radius of circle */
    }
   virtual void draw(); // to delay the identify of the draw
};
void circle_class::draw()   // Print
{  // details of draw circle class
  clearviewport(); /* clear View Point */
  circle(x1,y1,radius);  /* draw circle */
}
```

الجزء الـتــالي يـعــرف الكــائن arc_class حيــث يــرث مــن الصــنف circle_class الـدوال المنتمية ونـوع الـوراثـة عامة ويـشتمل علي الـدالـة المنتمية arc_class والمتغيرين stangle و enangle .

```
class arc_class: public circle_class    // Arc Class
{  // Drived From Circle_class
  private:
    int stangle; // define start angle
    int enangle; // define end angle
  public:
    arc_class(); // call   the function form arc class
    virtual void draw();  // use for draw the arc
};
arc_class :: arc_class()
{
  // circle_class :: circle_class();
  cout<<"\n\t"<<name<<" Enter The Start Angle = "; cin>>stangle;
  cout<<"\n\t"<<name<<" Enter The End   Angle =";   cin>>enangle;
}
void arc_class::draw()
{
  clearviewport();  // clear View Point
  arc(x1,y1,stangle,enangle,radius); // draw arc
}
```

الجزء التالي يعرف الصنف line_class حيث يرث من الصنف point_class ونوع الوراثة عامة ويشتمل علي المتغيرين x2 و y2 ويحتوي علي دالة البناء

```
class line_class : public point_class
{  // drived from point_class
  protected:
    int x2;  // define x point
    int y2;  // define y point
  public:
    line_class();  // constructor
    virtual void draw();  // draw function
}
line_class :: line_class()   // details of line class constructore
{  // get x2 and y2
cout<<"\n\t"<<name<<" Enter The Second X Coordinate (X2) = "; cin>>x2;
cout<<"\n\t"<<name<<" Enter The Second Y Coordinate (Y2) = "; cin>>y2;
}
void line_class:: draw()
{
    clearviewport(); // clear View Point
    line(x1,y1,x2,y2); /* draw line */
}
```

الجزء التالي يعرف الصنف trangle_class حيث يرث من الصنف السابق line_class ونوع الوراثة عامة ويشتمل علي المتغيرين x3 وy3 ويتضمن الدالة المنتمية - دالة البناء -

```
class trangle_class : public line_class
{  /* Drived From line class */
 protected:
   int x3; // define x3
   int y3; // define y3
 public:
   trangle_class();  // constructore
virtual void draw();
};
trangle_class:: trangle_class()

{   // get x3 ,y3
  cout<<"\n\t"<<name<<" Enter The Third X Coordinate (X3) = "; cin>>x3;
  cout<<"\n\t"<<name<<" Enter The Third Y Coordinate (Y3) = "; cin>>y3;
}
void trangle_class:: draw()   // detail
{
 clearviewport(); // clear View Point
 line(x1,y1,x2,y2); /* draw line */
 line(x1,y1,x2,y2); /* draw line */
 line(x2,y2,x3,y3); /* draw line */
 line(x3,y3,x1,y1); /* draw line */
}
```

الجزء التالي يعرف الصنف rectangular_class حيث يرث من الصنف trangle_class الدوال المنتمية ويشتمل علي المتغيرين x4 و y4 ويتضمن الدالة السطرية Inline Function – دالة البناء – وعلى الدالة الرسم

```
class rectangular_class  : public trangle_class
{  // Drived From trangle_class
 private:
     int x4; // define x4
     int y4; // define y4
 public:
      rectangular_class() // constructor
      {  /* inline function  */
       cout<<"\n\t"<<name<<" Enter The Forth X Coordinate (X4) = " ; cin>>x4;
       cout<<"\n\t"<<name<<" Enter The Forth Y Coordinate (Y4) = " ; cin>>y4;
      }
virtual void draw();  // polymorphism
};
void rectangular_class:: draw()
{
 clearviewport(); // clear View Point
 line(x1,y1,x1,y2); /* draw line */
 line(x1,y2,x2,y2); /* draw line */
 line(x2,y2,x2,y1); /* draw line */
 line(x2,y1,x1,y1); /* draw line */
}
```

الجزء التالي يعرف الصنف square_class الذي يرث من الكائن point_class ويشتمل علي المتغير d الذي يمثل الإزاحة وعلي الدالة السطرية ي دالة الرسم

```
class square_class : public point_class
{ // Derived From point class
private:
  int d; /* define edge variable */
public:
  square_class()
   {    // imline function
    cout<<"\n\t"<<name<<" Enter The Dimension = ";
    cin>>d; /* get the lenght of sequare edge */
   }
virtual void draw();
};
void square_class::draw()    // draw sequare
{
 clearviewport(); // clear View Point
 line(x1,y1,x1,y1+d); /* draw line for 1st edge */
 line(x1,y1+d,x1+d,y1+d); /* draw line for 2nd edge */
 line(x1+d,y1+d,x1+d,y1); /* draw line for 3th edge */
 line(x1,y1,x1+d,y1); /* draw line for 4th edge */
}
```

الجزء التالي يعرف الكائن cylinder_class الذي يرث من الصنف circle_class وراثة عامة ويشتمل علي المتغير offset وعلي دالة البناء ودالة الرسم draw .

```
class cylinder_class : public circle_class
{                 // Drived From Circle_class
  private:
    int offset;
  public:
    cylinder_class();  /* constructor */
virtual void draw(); // draw function as polymorphism
};
cylinder_class :: cylinder_class ()
{
 cout<<"\n\t"<<name<<" Enter The Offset = ";  // Offset length
 cin>>offset; // get the displacment
}
void cylinder_class::draw() /* Draw Cylinder */
{
  circle_class::draw(); /* call draw function */
  circle(x1,y1+offset,radius); /* draw cirle */
  line(x1+radius,y1,x1+radius,y1+offset); /* draw line */
  line(x1-radius,y1,x1-radius,y1+offset); /* draw line */
}
```

الجزء التالي يعرف بيئة الرسم والمتغيرات المتعلقة بها - أخذتها من ملف المساعدة الخاص بمترجم سي ++

```
void initialize() // initialize graphic enviroment
{
   /* request auto detection */
   int gdriver = DETECT, gmode, errorcode;
   /* initialize graphics and local variables */
   initgraph(&gdriver, &gmode, "");
   /* read result of initialization */
   errorcode = graphresult();
   /* an error occurred */
   if (errorcode != grOk)
   {
      cout<<"Graphics error: %s\n  ";
      cout<<grapherrormsg(errorcode);
      cout<<"Press any key to halt:";
      exit(1);
   }
}
```

الجزء التالي يعرف الدالة الفرعية main_menu والتي تظهر القائمة الرئيسية والتي تشتمل علي النقطة - الدائرة - الخط - المثلث - المربع - المستطيل - الاسطوانة .

```
void main_menu()
{
   clrscr();  // clear screen
   clearviewport();
   cout<<"\n\t########################Simple Graphic Program#####################";
   cout<<"\n\t# Main Menu                                                         #";
   cout<<"\n\t# =========                                                         #";
   cout<<"\n\t# [ 1 ] ----> Draw Arc                                              #";
   cout<<"\n\t# [ 2 ] ----> Draw Line                                             #";
   cout<<"\n\t# [ 3 ] ----> Draw Point                                            #";
   cout<<"\n\t# [ 4 ] ----> Draw Square                                           #";
   cout<<"\n\t# [ 5 ] ----> Draw Circle                                           #";
   cout<<"\n\t# [ 6 ] ----> Draw Trangle                                          #";
   cout<<"\n\t# [ 7 ] ----> Draw Cylinder                                         #";
   cout<<"\n\t# [ 8 ] ----> Draw Rectangular                                      #";
   cout<<"\n\t# [ 9 ] ----> Program Terminated                                    #";
   cout<<"\n\t###################################################################";
}
```

الجزء التالي يعرف الدالة الرئيسية main() حيث يطلب من المستخدم إدخال اسمة الشخصي لتصبح المخاطبة له بعد ذلك شخصية ، ثم يعرف المتغير shape مؤشر من نوع point_class ، أو بطريقة أخرى point_class *shape[4]; كمصفوفة .

```
int main()  /* main function part */
{
   int choice;
   cout<<"\n\n\n\n\n\n\n\n\t\t\t\t draw shapes" ;
   cout<<"\n\t\t\t Enter Your Name Please : ";
   cin.getline(name,30,'\n');
   initialize();       //For Graphic Initialization
   point_class *shape; // Declare an array of pointers to the base class property
   do
   {
   main_menu();  /* display the main menu */
   cout<<"\n\t\t\t"<<name<<" Enter Your choice = "; /* get your name */
   cin>>choice;
   switch(choice)  // Selection Part
    {
     case 1 :
              {
                shape=new arc_class;   // Draw Arc
                break;
              }
     case 2 :
              {

                shape=new line_class;    // Draw Line
                break;
              }
```

ثم يطلب من المستخدم اختيار الشكل المراد رسمه ، بإدخال الرقم المقابل له من الأرقام من 1 إلى 8 والرقم 9 للخروج من البرنامج .

```
  case 3 :
           {
             shape=new point_class;  // Draw Point
             break;
           }
  case 4 :
           {
             shape=new square_class();  // Draw Square
             break;
           }
  case 5 :
           {
             shape=new circle_class;     // Draw Circle
             break;
           }
  case 6 :
           {
             shape=new trangle_class;   // Draw Trangle
             break;
           }
  case 7 :
           {
             shape=new cylinder_class;   // Draw Cylinder
             break;
           }
```

```
                 ;
        case 8 :
                 {
                   shape=new rectangular_class;   // Draw Rectangular
                   break;
                 }
        case 9 :
                 {
                   break;
                 }
        default:
                 {
                   cout<<"\n\n\t\aInvalid Entry ...... Try Again";
                   getche();
                 }
       }  // End Switch
      if (choice < 9)
         {
           shape->draw();
           getche();
         }
    }   // End For while Loop
   while (choice!=9);
   delete shape;
   closegraph();    // Close Graph Libary
   return 0;
   }
```

- **جمل البرمجية بطريقة معطيات معرفة مسبقا – بشكل سريع**

```
class point_class     //  Base Class
{
 protected:
   int x1;
   int y1;
 public:

   point_class (int x11=150,int y11=150)  // Constructor
    {
      x1=x11;
      y1=y11;
    }
virtual void draw();    // Print Function
};
void point_class :: draw()
 {
  clearviewport();        // Clear View Point
  for(int i=1;i<=3;i++)
   {
    for(int j=0;j<=350;)     // Delay for 500ms
      {
       putpixel(x1+j,y1,i+12);
       for(long c=1;c<=8000000;c++)  { }
       j=j+50;
      }
       y1=y1+50;    // Next
   }
 }
class circle_class : public point_class  // Circle Class
{                    // Drived From point_class
 protected:
   int radius;
 public:
   circle_class(int x11=300,int y11=150,int rad=45);
   virtual void draw();
};
circle_class:: circle_class(int x11,int y11,int rad): point_class(x11,y11)
{
 radius=rad;
}
```

```
void circle_class::draw()        // Print
{

  clearviewport();                  // Clear View Point
  circle(x1,y1,radius);
}
class arc_class: public circle_class    // Arc Class
{              // Drived From Circle_class
  private:
    int stangle;
    int enangle;
  public:
    arc_class(int xx=100,int yy=150,int rad=45,int start=0,int end=90);
virtual void draw();
};
arc_class :: arc_class (int x11,int y11,int rad,int start,int end)
                        :circle_class(x11,y11,rad)
{
  stangle=start;
  enangle=end;
}
void arc_class::draw()
{
  clearviewport();
  arc(x1,y1,stangle,enangle,radius);
}
class line_class : public point_class
{             // Drived From point_class
  protected:
    int x2;
    int y2;
  public:
    line_class(int x1=70,int y1=0,int x22=500,int y22=500);
virtual void draw();
}
line_class :: line_class(int x1,int y1,int x22,int y22)
                           : point_class(x1,y1)
{
  x2=x22;
  y2=y22;
}
void line_class:: draw()
{

     clearviewport();
     line(x1,y1,x2,y2);
}
class trangle_class : public line_class
{               // Drived From line class
 protected:
   int x3;
   int y3;
 public:
   trangle_class(int x11=110,int y11=120,int x22=300,
         int y22=400,int x33=450,int y33=300);
virtual void draw();
};
trangle_class:: trangle_class(int x11,int y11,int x22,int y22,int x33,int y33)
                                      :line_class(x11,y11,x22,y22)
{
  x3=x33;
  y3=y33;
}
void trangle_class:: draw()
{
 clearviewport();
 line(x1,y1,x2,y2);
 line(x1,y1,x2,y2);
 line(x2,y2,x3,y3);
 line(x3,y3,x1,y1);
}
class rectangular_class  : public trangle_class
```

```
{                // Drived From trangle_class
 private:
     int x4;
     int y4;
 public:
      rectangular_class(int x11=100,int y11=100,int x22=100,int y22=400,
                int x33=500,int y33=100,int x44=500,int y44=400);
virtual void draw();
};
rectangular_class :: rectangular_class(int x11,int y11,int x22,int y22,int x33,int y33,int
x44,int y44)
                    :trangle_class(x11,y11,x22,y22, x33, y33)
{
  x4=x44;
  y4=y44;
}
void rectangular_class:: draw()
{
 clearviewport();
 line(x1,y1,x1,y2);
 line(x1,y2,x2,y2);
 line(x2,y2,x4,y4);
 line(x4,y4,x3,y3);
 line(x3,y3,x1,y1);
}
class square_class : public point_class
{                   // Derived From point class
private:
  int d;
public:
  square_class(int x11=220,int y11=150,int d=150);
virtual void draw();
};
square_class :: square_class(int x11,int y11,int distance)
                      : point_class (x11,y11)
{
  d=distance;
}
void square_class::draw()    // draw sequare
{
 clearviewport();
 line(x1,y1,x1,y1+d);
 line(x1,y1+d,x1+d,y1+d);
 line(x1+d,y1+d,x1+d,y1);
 line(x1,y1,x1+d,y1);
}
class cylinder_class : public circle_class
{                // Drived From Circle_class
  private:
    int offset;
  public:
    cylinder_class(int x11=300,int y11=150,int  rad=45,int offs=150);
virtual void draw();
};
cylinder_class :: cylinder_class (int x11,int y11,int rad,int offs)
                             : circle_class (x11,y11,rad)
{
offset=offs;  // Offset length
}
void cylinder_class::draw()    // Draw Cylinder
{
  circle_class::draw();
  circle(x1,y1+offset,radius);
  line(x1+radius,y1,x1+radius,y1+offset);
  line(x1-radius,y1,x1-radius,y1+offset);
}
void initialize()
{
   /* request auto detection */
   int gdriver = DETECT, gmode, errorcode;
   /* initialize graphics and local variables */
   initgraph(&gdriver, &gmode, "");
   /* read result of initialization */
```

```
    errorcode = graphresult();
    /* an error occurred */
    if (errorcode != grOk)
    {
       cout<<"Graphics error: %s\n  ";   cout<<grapherrormsg(errorcode);
       cout<<"Press any key to halt:";
       exit(1);
    }
 }
 arc_class a;
 line_class l;
 point_class p;
 circle_class c;
 square_class s;
 trangle_class t;
 cylinder_class y;
 rectangular_class r;
 void main_menu()
 {
    clrscr();       // Clear Screen
    clearviewport();
    cout<<"\n\t\t\t\tSimple Graphic Program";
    cout<<"\n\t\t\t\t======================";
    cout<<"\n\n\tMain Menu";
    cout<<"\n\t=========";
    cout<<"\n\t[ 1 ] ----> Draw Arc";
    cout<<"\n\t[ 2 ] ----> Draw Line";
    cout<<"\n\t[ 3 ] ----> Draw Point";
    cout<<"\n\t[ 4 ] ----> Draw Square";
    cout<<"\n\t[ 5 ] ----> Draw Circle";
    cout<<"\n\t[ 6 ] ----> Draw Trangle";
    cout<<"\n\t[ 7 ] ----> Draw Cylinder";
    cout<<"\n\t[ 8 ] ----> Draw Rectangular";
    cout<<"\n\t[ 9 ] ----> Program Terminated";
    cout<<"\n\n\tEnter Yoiur choice = ";
 }
 int main()
 {
    int choice;
    initialize();            //For Graphic Initialization
    do
    {
    main_menu();
    cin>>choice;
    switch(choice)           // Selector Part
    {
      case 1 : { a.draw();   break;  }  // Draw Arc
      case 2 : { l.draw();   break;  }  // Draw Line
      case 3 : { p.draw();   break;  }  // Draw Point
      case 4 : { s.draw();   break;  }  // Draw Square
      case 5 : { c.draw();   break;  }  // Draw Circle
      case 6 : { t.draw();   break;  }  // Draw Trangle
      case 7 : { y.draw();   break;  }  // Draw Cylinder
      case 8 : { r.draw();   break;  }  // Draw Rectangular
      default :{
               if (choice !=9)
                   cout<<"\n\t\aInvalid Choice ...Try Again.";
               else
                   cout<<"\n\tGooD Bye.";
              }
    }  // End Case
  getche();
 } // Do
  while (choice!=9);    // Condition to Break
  closegraph();
  return 0;
 }
```

- ملاحظة :-

1. لا يوجد صعوبة في إعداد وتكوين البرنامج ، فقط الذي يحتاجه المبرمج هو معرفة التامة بعملية الوراثة والاشتقاق وذلك يتأتى من خلال الممارسة والتدريب .
2. إذا كانت الدالة المراد استدعاؤها " دالة بناء " للصنف فانه لا داعي من كتابة Prototype خاص بها ، ولكن إذا كانت تشكل إحدى الدوال المنتمية فيجب استدعاؤها بوضع نموذج خاص لها في الدالة المشتقة .

- **عدد ساعات العمل :-**

- بناء البرمجية = 3 ساعة .
- كتابة التقرير = 4 ساعة .

- **المقترحات والتطورات :-**

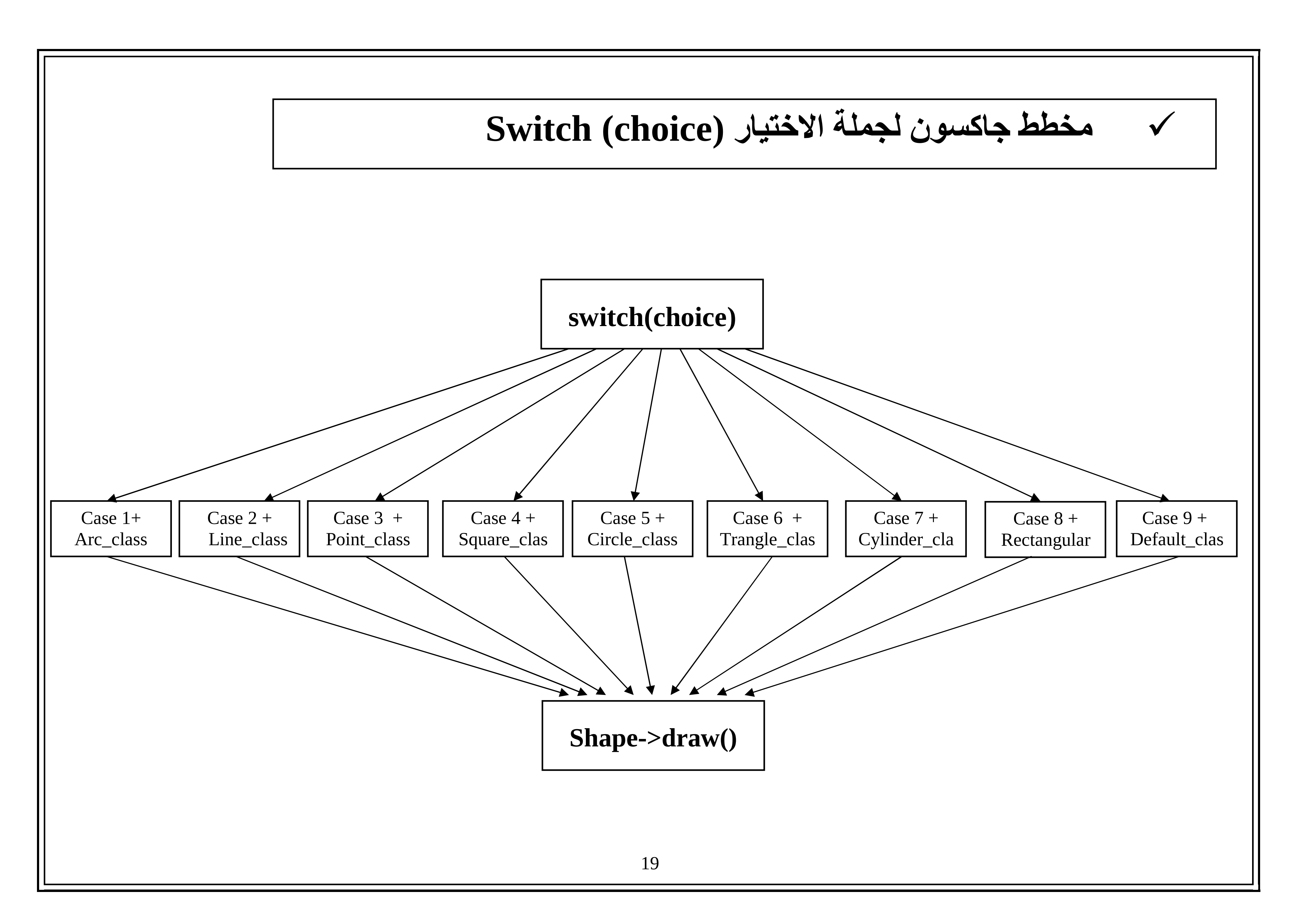
تتمثل في بناء برمجية تشابه في عملها برنامج الرسام الذي يأتي مع حزمة نظام التشغيل windows **، وذلك باستخدام لغة البرمجة** Microsoft Visual C++ **حيث أنها لغة رائعة جدا ، وكثير أحببتها ، وكان في تخطيطي أن اعمل إما برنامج آلة حاسبة ، أو برنامج الرسام من خلال هذه اللغة – ولكن للآسف لم يكن لدي الوقت الكافي لتعلم هذه اللغة كما يجب – وادعوا زملائي الطلبة أن يتجهوا إلى هذه اللغة وانصحهم أن يفهموا لغتي سي و سي ++ لانهما هما الأساس .**

الرسام

- **المشاكل والصعوبات :-**

المشكلة التي يستحق ذكرها هي أن جهازي يعمل مرة ويرتاح مرة أخرى ، وبالنسبة لمشاكل التصميم البرنامج واجهت مشكلة في حجز موقع باستخدام المؤشرات ولكن تم حلها بتوفيق الله .

{ تم بحمد الله }


✓ مخطط جاكسون لجملة الاختيار Switch (choice)
switch(choice)
Case 1+
Arc_class
Case 2 +
Line_class
Case 3 +
Point_class
Case 4 +
Square_clas
Case 5 +
Circle_class
Case 6 +
Trangle_clas
Case 7 +
Cylinder_cla
Case 8 +
Rectangular
Case 9 +
Default_clas
Shape->draw()